REMARKS

Claims 1, 2, and 21 stand rejected under 35 USC §103(a) as being unpatentable over Prologol et al., U.S. patent 6,823,478 in view of Ashley et al., U.S. patent application 2004/0088142. Claims 3-5 and 8-12 stand rejected under 35 USC §103(a) as being unpatentable over Prologol et al., U.S. patent 6,823,478 in view of Ashley et al., U.S. patent application 2004/0088142 and further in view of Juettner et al., U.S. patent 5,615,333. Claims 6 and 7 stand rejected under 35 USC §103(a) as being unpatentable over Prologol et al., U.S. patent 6,823,478 in view of Ashley et al., U.S. patent application 2004/0088142 and further in view of Juettner et al., U.S. patent 5,615,333 and Wygodny et al., U.S. patent 5,657,438. Claim 12 stands rejected under 35 USC §103(a) as being unpatentable over Prologol et al., U.S. patent 6,823,478 in view of Ashley et al., U.S. patent application 2004/0088142 and further in view of Juettner et al., U.S. patent 5,615,333 and Gross et al., U.S. patent 5,371,883. Claims 13, 14 and 17-19 stand rejected under 35 USC §103(a) as being unpatentable over Juettner et al., U.S. patent 5,615,333 in view of Prologol et al., U.S. patent 6,823,478. Claims 15 and 16 stand rejected under 35 USC §103(a) as being unpatentable over Juettner et al., U.S. patent 5,615,333 in view of Prologol et al., U.S. patent 6,823,478 in view of Wygodny et al., U.S. patent 5,657,438. Claim 20 stands rejected under 35 USC §103(a) as being unpatentable over Juettner et al., U.S. patent 5,615,333 in view of Prologol et al., U.S. patent 6,823,478 in view of Gross et al., U.S. patent 5,371,883.

Claims 1, 8, 10, 11, 12, 13, 14, 15, 16, 17, and 21 have been amended to more clearly state the invention. Reconsideration and allowance of each of the claims

1-21, as amended, is respectfully requested.

Prologol et al., U.S. patent 6,823,478 discloses a system and method for automating the testing of processing environment changes. Input data having corresponding known-good output based on the processing of the input data in a first state of a processing environment is received by a comparison mechanism. The comparison mechanism processes the input data in a changed processing environment as compared with the processing environment in the first state. The comparison mechanism automatically evaluates the generated output against the known-good output to identify differences between the generated output and the known-good output. If no differences are found between the generated output and the known-good output, the comparison mechanism stores the generated output as the known-good output. The comparison mechanism may also generate an error log if there are differences between the generated output and the known-good output, generate an email message including the error log, and transmit the email message to a tester responsible for evaluating the error log. The generated output, the known-good output, and the error log may be formatted in an extensible markup language format. In addition, the comparison mechanism may test the generated output for compliance with another format, such as an electronic filing format.

Ashley et al., U.S. patent application 2004/0088142 discloses a customer configuration server may provide access to configuration data for multiple computer systems and organize the configuration data for the computer systems within a hierarchy of logical groups. Logical groups may correspond to geographic locations,

customers, and individual computer systems. For example, configuration data for several computer systems may be logically grouped by customer, and configuration data for several customers may be logically grouped by geographic location. The configuration data may be available from a variety of different data sources and accessible via a single software application. Such a software application may also provide an interface to various common functions, allowing those functions to be applied to any of the computer systems' configuration data.

[0071] In some embodiments, an application server 235 may include a remote patch server. For example, a remote patch server may be dedicated to provide patches to a customer (e.g., a patch tar bundle of a new version of an explorer) or to another application (e.g., a patch tar bundle of a new version of customer configuration server 100). In one embodiment, the remote patch application server 235 may be sent a file output name, email address(es) of user(s) which should receive the patch bundle, and a list of which patches to include in the patch bundle. The remote patch server may respond by sending a size of the patch bundle to the requesting user and emailing the patch to the specified addresses. If another patch server is located closer to the requesting user, the patch server that receives the request for a patch may transfer the request to the other patch server, which may in turn provide the requested patch bundle to the requesting user.

Juettner et al., U.S. patent 5,615,333 discloses a method for integration testing of all objects of an object-oriented program in which an analysis of its source code is first carried out in order to identify the mutual dependencies between objects

and classes and, on the basis of these dependencies, those objects that have already been tested are allocated to a set of tested objects and, further, objects that are dependent only on these tested objects are also identified. These objects are tested and are subsequently added to the set of tested objects. When no object can be found that is dependent only on tested objects, those objects are sought that are dependent only on one additional, untested object. This untested object can then be replaced by an object stub for test purposes. The object thus tested is subsequently transferred into the set of tested objects. When objects are found that are dependent only on one another and form a cycle, then one of these is replaced by an object stub in order to be able to test the other. As a result, all objects of the object-oriented program to be tested are united in the set of tested objects and the entire program is tested.

Wygodny et al., U.S. patent 5,657,438 discloses a system for developing tests of a System Under Test (SUT) which includes a Central Processing Unit (CPU), a screen and input apparatus. The system for developing tests includes a manipulation apparatus enabling an operator to manipulate, within a test workspace, a sequence of test script statements into a desired script, wherein the test script statements describe operator commands to the SUT and screen capture and verify operations and b) interactive execution apparatus for executing at least a portion of the desired script by providing the at least a portion of the desired script to the SUT thereby to operate the SUT as desired. Column 2, line 64 - column 3, line 5 states: Still further, in accordance with an embodiment of the present invention, the editor apparatus includes apparatus for providing an execution marker indicating a currently executed test script statement

and the execution apparatus includes apparatus for indicating to the editor apparatus to move the execution marker to a next test script statement to be executed. The editor apparatus includes execution marker moving apparatus enabling a user to move the execution marker to a desired next test script statement to be executed.

Gross et al., U.S. patent 5,371,883 discloses an improved method of testing in a distributed environment which is comprised of a Control Program residing in a Control Machine. The Control Machine also contains the central repository of information to control test execution in the Test Machines. The Control Program forwards instructions to a particular Test Program, residing in a Test Machine. The instructions are executed on that machine, and results are reported back to the Control Program. The Control Program verifies whether the results are correct. Depending on the results of the verification, the Control Program sends the test machine further instructions (to continue the test, stop the test, etc.). Logging the results of each test operation, keeping track of the tests performed, and coordinating the test cases are all performed on the Control Machine, by the Control Program. Column 7, lines 58-68 states: The above-described invention provides a more convenient technique for testing programs in a distributed environment. As discussed above, by using the central repository of test cases and their expected results, the invention avoids the problems of keeping track of and coordinating differing test cases. It also provides a convenient means of executing multi-station testing, where the interaction of multiple test machines is at issue. Finally, by having all test results on a single system, it makes simple the creation of reports comparing the results of the various systems and tests.

Reconsideration and allowance of amended claims 1, 13, and 21, as amended, is respectfully requested.

The present invention solves problems with the use of conventional software fix programs. After a software product is released, the producer may provide interim updates or fixes for the software product before the next release of the software product. These updates or fixes between releases may include program temporary fixes (PTFs). On some systems, program temporary fixes (PTFs) may be sent out to customers as group PTFs or fix packs. These fix packs often contain multiple PTFs. At times a PTF is shipped which resolves one problem but introduces another problem. It can be extremely difficult to isolate which PTF causes the new problem, and the usual approach is for a programmer to debug the new problem for the overall software product.

Independent claim 1, as amended, recites a method for implementing autonomic testing and verification of software fix programs comprising the steps of: receiving a software fix program; said software fix program including multiple patches; sequentially applying each patch of said multiple patches of said software fix program to a software product; testing said software product responsive to each said sequentially applied patch; providing test results to a user responsive to said testing of said software product; sequentially applying iterations of each patch of said multiple patches of said software fix program to a software product and different combinations of said patches to the software product; calling a test program and receiving expected test results for each applied iteration to the software product; testing said software product for each applied

iteration to the software product; and comparing test results to the expected test results for each applied iteration to the software product; and saving test results for each applied iteration to the software product and displaying said test results to the user

Applicants respectfully submit that independent claim 1, as amended, is patentable over all of the references of record.

Applicants respectfully submit that the Prologo et al., Ashley et al., Juettner et al., Gross et al., and Wygodny et al. references do not enable, nor provide any suggestion of receiving a software fix program; said software fix program including multiple patches; and sequentially applying each patch of said multiple patches of said software fix program to a software product, and further sequentially applying iterations of each patch of said multiple patches of said software fix program to a software product and different combinations of said patches to the software product; calling a test program and receiving expected test results for each applied iteration to the software product; testing said software product for each applied iteration to the software product. The references of record fail to suggest receiving a software fix program; said software fix program including multiple patches; and sequentially applying each patch of said multiple patches of said software fix program to a software product, and further sequentially applying iterations of each patch of said multiple patches of said software fix program to a software product and different combinations of said patches to the software product, as recited in independent claim 1, as amended.

The references of record do not enable, nor provide any suggestion of testing said software product responsive to each said sequentially applied patch; and

providing test results to a user responsive to said testing of said software product, and calling a test program and receiving expected test results for each applied iteration to the software product; testing said software product for each applied iteration to the software. Thus, independent claim 1, as amended, is patentable.

Independent claim 13, as amended, recites apparatus for implementing autonomic testing and verification of software fix programs comprising: an isolation manager for receiving a software fix program containing a plurality of patches; a user interface coupled to said isolation manager ~~for~~ receiving user input selections and reporting results to a user; said isolation manager sequentially applying each patch of said plurality of patches to a software product; and testing said software product responsive to each said sequentially applied program; and providing test results to a user responsive to said testing of said software product; and said isolation manager sequentially applying iterations of each patch of said multiple patches of said software fix program to the software product and different combinations of said patches to the software product; calling a test program and receiving expected test results for each applied iteration to the software product; testing said software product for each applied iteration to the software product; and comparing test results to the expected test results for each applied iteration to the software product; and saving test results for each applied iteration to the software product and displaying said test results to the user.

Applicants respectfully submit that independent claim 13, as amended, is patentable over the references of record.

Applicants respectfully submit that the references of record do not enable,

nor provide any suggestion of an isolation manager sequentially applying iterations of each patch of said multiple patches of said software fix program to the software product and different combinations of said patches to the software product; calling a test program and receiving expected test results for each applied iteration to the software product; testing said software product for each applied iteration to the software product; and comparing test results to the expected test results for each applied iteration to the software product; and saving test results for each applied iteration to the software product and displaying said test results to the user, as taught and claimed by Applicants. Thus, independent claim 13, as amended, is patentable.

Independent claim 21, as amended, recites a computer program product for implementing autonomic testing and verification of software fix programs. Independent claim 21, as amended, recites the steps of receiving a software fix program software fix program including multiple patches; sequentially applying each patch of said multiple patches of said software fix program to a software product; testing said software product responsive to each said sequentially applied patch; providing test results to a user responsive to said testing of said software product; sequentially applying iterations of each patch of said multiple patches of said software fix program to a software product and different combinations of said patches to the software product; calling a test program and receiving expected test results for each applied iteration to the software product; testing said software product for each applied iteration to the software product; and comparing test results to the expected test results for each applied iteration to the software product; and saving test results for each

applied iteration to the software product and displaying said test results to the user

Independent claim 21, as amended, is patentable over all the references of record. Independent claim 21, as amended, is patentable for the same reasons as independent claim 1.

Applicants respectfully submit that the references of record do not enable, nor provide any suggestion a computer program product for implementing autonomic testing and verification of software fix programs, as taught by Applicants and claimed in independent claim 21, as amended.

Thus, independent claim 21, as amended, is patentable.

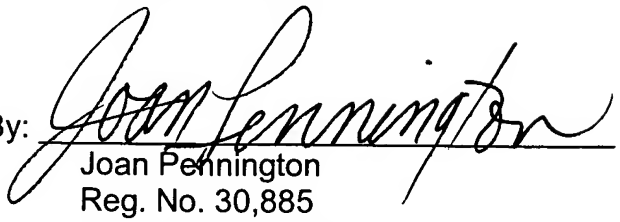
Dependent claims 2-12, and 14-20, as amended, depend from patentable claims 1 and 13, further define the subject matter of the invention, and each of the dependent claims 2-12, and 14-20, as amended, is patentable.

Applicants have reviewed all the art of record, and respectfully submit that the claimed invention is patentable over all the art of record, including the references not relied upon by the Examiner for the rejection of the pending claims.

It is believed that the present application is now in condition for allowance and allowance of each of the pending claims 1-21, as amended, is respectfully requested. Prompt and favorable reconsideration is respectfully requested.

If the Examiner upon considering this amendment should find that a telephone interview would be helpful in expediting allowance of the present application, the Examiner is respectfully urged to call the applicants' attorney at the number listed below.

Respectfully submitted,

By: 

Joan Pennington
Reg. No. 30,885
Telephone: (312) 670-0736